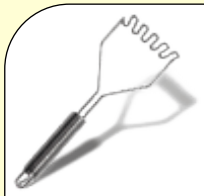
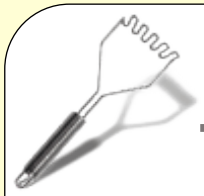
Mashups: Remixing the Web

Lecture 9: Introduction to jQuery



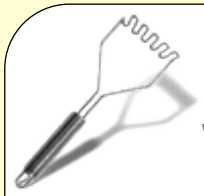
Reminders

- Assignment #4 due next week
- Project Checkpoint #1 next week (no hand-in)
- Office hours on **Tuesday** next week
- Updates on Assignment #4
  - Latest assignment handout has correct SMS number
  - Assignment vs. Project: What do I work on?
- Next week:
  John Britton guest lecture on Twilio API



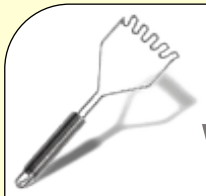
Today's Goals

- Quick intro to some useful JavaScript libraries: jQuery and Processing.js
- Learn how to use jQuery to select elements, make AJAX calls, and animate objects
- Learn how to draw with Processing.js
- Q&A on Assignment #4
- Meet with project teams to discuss progress



What is jQuery?

- jQuery is a JavaScript library.
- It provides a set of alternative ways to accomplish common programming tasks.
- It simplifies some things that are typically cumbersome in JavaScript.
- It is an open source project, maintained by a group of developers, with a very active support base and thorough, well written documentation.



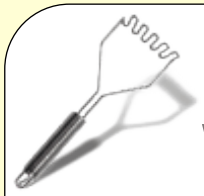
What jQuery is not…

- jQuery is not a substitute for knowing JavaScript.

  jQuery is very useful, but you should still know how JavaScript works and how to use it correctly. That's why we learned basic JavaScript first!

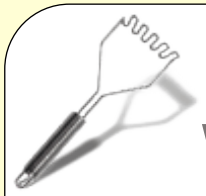
- jQuery is not a solution for everything.

  There is still plenty of worthwhile functionality built into JavaScript! Don't turn every project into a quest to "jQuery-ize" the problem. Use jQuery where it makes sense.



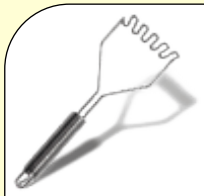
What is available with jQuery?

- Cross browser support and detection
- AJAX functions
- CSS functions
- DOM manipulation
- DOM transversal
- Attribute manipulation
- Event handling
- JavaScript animation
- Hundreds of plug-ins for pre-built user interfaces, advanced animations, form validation, etc.



Why jQuery?

- Lightweight – only 14KB (Minified and Gzipped)
- Cross-browser support
  (IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+)
- CSS-like syntax – easy to understand
- Active developer community
- Extensible through plug-ins



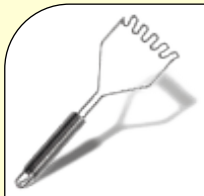
Getting Started with jQuery

- Reference it in your markup:

```
<script src="jquery.js"/>
```

- Or reference it from Google:

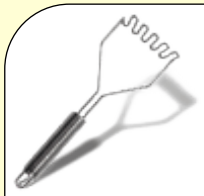
```
<script src="http://ajax.googleapis.com/
             ajax/libs/jquery/1.3.2/
             jquery.min.js">
</script>
```



It's All About the Magic $

- Create HTML elements on the fly

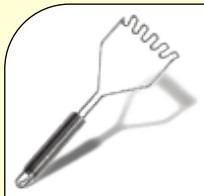
```
var el = $("<div/>")
```



It's All About the Magic $

- Manipulate existing DOM elements

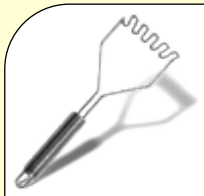
```
$(window).width()
```



It's All About the Magic $

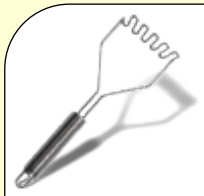
- Select document elements

```
$("div").hide();

$("div", $("p")).hide();
```



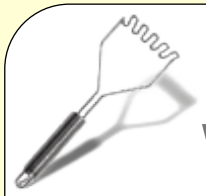
Selecting Elements the Typical Way

```
document.getElementById("menu")
document.getElementsByTagName("p")
```



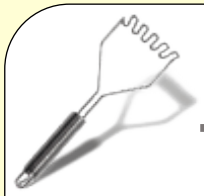
Selecting Elements the jQuery Way

```
document.getElementById("menu")
document.getElementsByTagName("p")

$("#menu")
$("p")
```



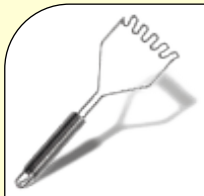
What's Going on Here?

- Selecting part of document is a fundamental operation in jQuery.
- A JQuery *object* is a wrapper for a selected group of DOM nodes.
- The `$()` function is a factory method that creates JQuery objects.
- `$("p")` is a JQuery object containing all of the `<p>` elements in the document.



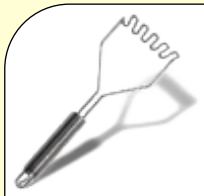
The jQuery ($) Object

- `$` is used with all `jQuery` functions.
- `$` is actually a shortcut for `jQuery`, which you can use instead if you prefer.
- The `$` by itself represents the `jQuery` object. When combined with a *selector*, can represent multiple DOM Elements, as shown on the next slides.



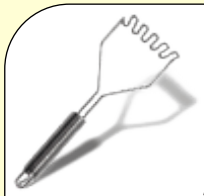
jQuery CSS Selectors

- `p` element name
- `#id` identifier
- `.class` classname
- `p.class` element with class
- `p a` anchor as any descendant of p
- `p > a` anchor direct child of p



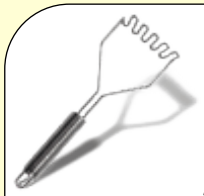
jQuery Selection Filters

- `p:first` — first paragraph
- `li:last` — last list item
- `a:nth(3)` — fourth link
- `a:eq(3)` — fourth link
- `p:even` or `p:odd` — every other paragraph
- `a:gt(3)` or `a:lt(4)` — every link after the fourth or up to the fourth
- `a:contains('click')` — links that contain the word “click”



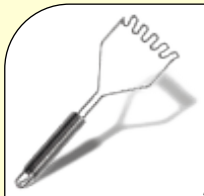
Adding Classes with jQuery

- The `.addClass()` method changes the selected DOM nodes by changing their 'class' attribute
- `$("dt").addClass("emphasize")` will change all occurrences of `<dt>` to `<dt class="emphasize">`
- Use `.removeClass()` to remove classes



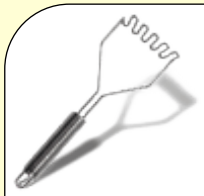
Adding a Class the Typical Way

```
function hasClass (ele,cls) {
  return ele.className.match(
    new RegExp('(\\s|^)' + cls + '(\\s|$)'));
}
function addClass (ele,cls) {
  if (!hasClass(ele,cls)) {
    ele.className += " " + cls;
  }
}
function getElementsByClass(searchClass,node,tag) {
  ...
}
var expandos = getElementsByClass('wd-expando');
for (var i = 0; i < expandos.length; i++) {
  var expando = expandos[i];
  addClass(expando, 'wd-expando-on');
}
```



Adding a Class the jQuery Way

```
$('.wd-expando').addClass('wd-expando-on');
```



jQuery Effects

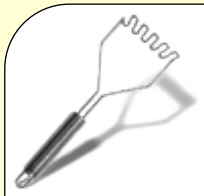
```
.css('property', 'value')
.css({'prop1':'value1','prop2':'value2'…})
```

For example:

```
  .css('color', 'red')
```

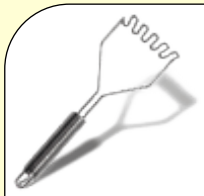
```
.hide(speed) or .show(speed)
```

where speed is `'slow'` or `'normal'` or `'fast'`



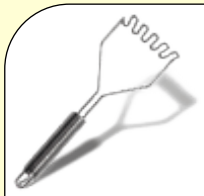
Show/Hide the Typical Way

```
<div id="more"></div>
<a href="#" onclick="toggle_visibility('more');">
  Click here to toggle visibility of #more</a>

function toggle_visibility(id) {
  var e = document.getElementById(id);
  if(e.style.display == 'block') {
    e.style.display = 'none';
  } else {
    e.style.display = 'block';
  }
}
```



Show/Hide the jQuery Way

```
<div id="more"></div>
<a href="#">
  Click here to toggle visibility of #more</a>

$("a").click(function(){
  $("#more").toggle("slow");
  return false;
});
```



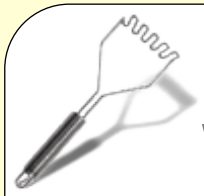
jQuery Events

- We just saw an example of adding an event handler to a DOM element.
- The general syntax for adding events is:

```
.bind(eventname, function)
```

  - `'click'`
  - `'change'`
  - `'resize'`

-
```
$("a").bind('click',function(){
  $(this).addClass('red');});
```

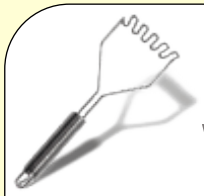


Waiting Until the Page Loads

- Typically, we would use the onLoad event to run something once the page has finished loading:

```
function doBold (){$("p").addClass("bold")}
<body onLoad="doBold()">
```

- We had to alter the HTML. This is bad. Remember that structure and appearance should be separated!
- Also, `onLoad` waits until all images etc are loaded. We really only need to wait until the DOM has loaded.



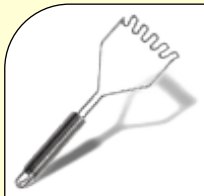
Waiting Until Load, the jQuery Way

- JQuery provides an independent scheduling point after the DOM is created, but before the images are loaded:

```
$(document).ready(doBold);
```

- Now there are no HTML changes required – everything is in the done in script.
- Typical jQuery pattern:

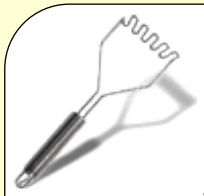
```
$(document).ready(function(){
  $("p").addClass("bold")
});
```



Chaining Functions Together

JQuery supports function *chaining*, in which the output of one function call becomes the input of the next.

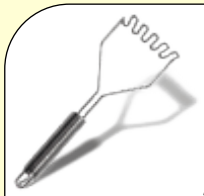
```
$('a:contains("home")').
   parent().
   addClass("emphasize")
```



All-Purpose jQuery Syntax

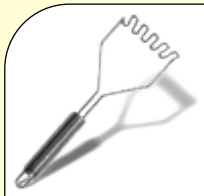
```
$(selector).func1(…).func2(…).funcN(…);
```

`$`	jQuery Object (you can also use `jQuery`)
`selector`	Selector syntax, many different selectors allowed
`func`	Chainable, most functions return a jQuery object
`(…)`	Function parameters



A Note on jQuery Functions

- jQuery functions are either attached to the jQuery object ($) or chained off of a selector statement.
- Most functions return the jQuery object they were originally passed, so you can perform many actions in a single line.
- The same function can perform an entirely different action based on the number and type of parameters.



Changing the DOM with jQuery

`.attr({'name', 'value'})`

Sets a new attribute (or multiple attributes).

`$('<i>hello</i>')`

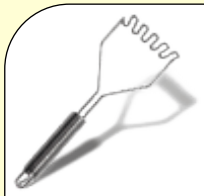
Creates a new element.

`$('<i>hello</i>').insertAfter('div.main p');`

Creates an element and inserts it into the document.

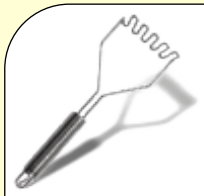
`.html()` or `.text()` or `.empty()`

Replace matched elements with newly created elements.



Complete List of jQuery Selectors

- `$( html )`
  Create DOM elements on-the-fly from the provided String of raw HTML.
- `$( elems )`
  Wrap jQuery functionality around single or multiple DOM elements.
- `$( fn )`
  A shorthand for `$(document).ready()`, allowing you to bind a function to be executed when the DOM document has finished loading.
- `$( expr, context )`
  This function accepts a string containing a CSS or basic XPath selector which is then used to match a set of elements. Default context is document. Used most often for DOM transversal.
- Selectors will return a jQuery object, which can contain one or more elements, or contain no elements at all.



jQuery Selector Examples

- **$(html)**

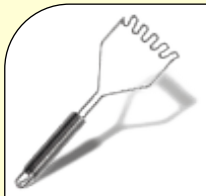
```
$('<p><a href="index.html">
   Click here!</a></p>')
```

- **$ (elems)**

```
$(document), $(window), $(this)
$(document.getElementsByTagName('p'))
```

- **$ (fn)**

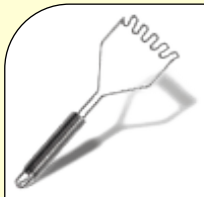
```
$(function() { alert('Hello, World!') });
```



jQuery Selector Examples

- **$ (expr, context)**

```
$("p"), $("form"), $("input")
$("p#content"), $("#content"), $(".brandnew"),
  $("p span.brandnew:first-child, #content")
$("p/span"), $("p/span[@class=brandnew]"),
  $(p/span:first), $(p:first/span:even)
$("input:checkbox[@checked]"),
  $("div:visible p[a]")

var xml = '<d><it w="h1"><nm>One</nm></it><it
  w="h2"><nm>Two</nm></it></d>';
$("d it nm:contains('One')", xml),
  $("it[@w^=h]",xml)
```



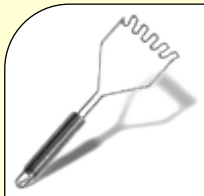
jQuery Example

```
$("li:odd").prepend('<span>Changed</span>').css({background:"red"});
```

```
<ul>
  <li>
    First item
  </li>
  <li>
    Second item
  </li>
  <li>
    Third item
  </li>
</ul>
```

```
<ul>
  <li>
    <span>Changed</span>
    First item
  </li>
  <li>
    Second item
  </li>
  <li>
    <span>Changed</span>
    Third item
  </li>
</ul>
```

```
<ul>
  <li style="background:red;">
    <span>Changed</span>
    First item
  </li>
  <li>
    Second item
  </li>
  <li style="background:red;">
    <span>Changed</span>
    Third item
  </li>
</ul>
```



More Complex jQuery Example

```
$("div:hidden").find(".foo").empty().text("Changed").end().show();
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Goodbye cruel world.
  </span>
</div>
```

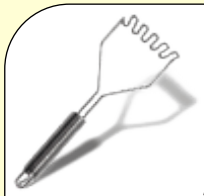
```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Goodbye cruel world.
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Changed
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:none">
  <span>
    More text
  </span>
  <span class="foo">
    Changed
  </span>
</div>
```

```
<div>
  <span class="foo">
    Some text
  </span>
</div>
<div style="display:block">
  <span>
    More text
  </span>
  <span class="foo">
    Changed
  </span>
</div>
```



Advanced jQuery Example

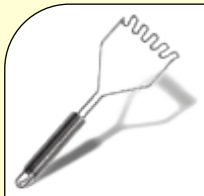
```
$("span.none").click(
  function(){
    $(this).siblings(":checkbox").removeAttr("checked");
  }
);
```

```
$("span.all").click(
  function(){
    $(this).siblings(":checkbox").attr("checked","checked");
  }
);
```

or

```
$("span").click(
  function(){
    if($(this).text()=="Select All"))
    $(this).siblings(":checkbox").attr("checked","checked");
    else if ($(this).attr("class")=="none")
    $(this).siblings(":checkbox").removeAttr("checked");
  }
);
```

```
<div>
  <span class="all">Select All</span>
  <span class="none">Select None</span>
  <input name="chk1" type="checkbox"/>
  <input name="chk2" type="checkbox"/>
  <input name="chk3" type="checkbox"/>
</div>
<div>
  <span class="all">Select All</span>
  <span class="none">Select None</span>
  <input name="chk4" type="checkbox"/>
  <input name="chk5" type="checkbox"/>
  <input name="chk6" type="checkbox"/>
</div>
```

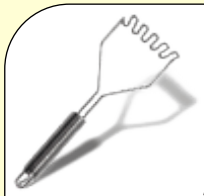


jQuery and AJAX

- jQuery has a set of functions that provide a common interface for AJAX, no matter what browser you are using.
- Most of the AJAX functions are called using the same general pattern:

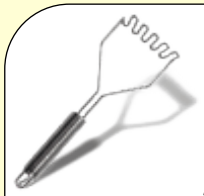
```
$.func(url[,params][,callback])
```

  - `url` URL of server target
  - `params` (optional) names and values to send to server
  - `callback` (optional) function executed after successful communication.



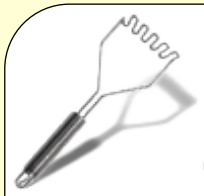
AJAX Calls the Typical Way

```
var xmlHttp;
if (window.XMLHttpRequest) {
  xmlHttp = new window.XMLHttpRequest();
}
else { // Internet Explorer 5/6
  xmlHttp =
    new ActiveXObject("Microsoft.XMLHTTP");
}
xmlHttp.open('GET',
  "albumcover.php?artist=cher&album=believe", true);
xmlHttp.onreadystatechange = function() {
  processCover(xmlHttp);
};
xmlHttp.send(null);
```



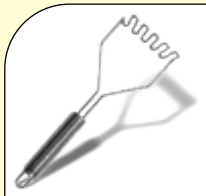
AJAX Calls the jQuery Way

```
$.get("albumcover.php",
  {artist: "Cher", album: "Believe"},
  processCover);
```



Complete List of jQuery AJAX Functions

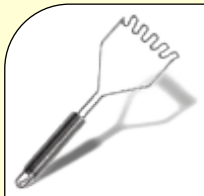
- **$.func(url[,params][,callback])**
  - $.get
  - $.getJSON
  - $.getIfModified
  - $.getScript
  - $.post
- **$(selector), inject HTML**
  - load
  - loadIfModified
- **$(selector), ajaxSetup alts**
  - ajaxComplete
  - ajaxError
  - ajaxSend
  - ajaxStart
  - ajaxStop
  - ajaxSuccess
- **$.ajax, $.ajaxSetup**
  - async
  - beforeSend
  - complete
  - contentType
  - data
  - dataType
  - error
  - global
  - ifModified
  - processData
  - success
  - timeout
  - type
  - url



jQuery AJAX Example

```
<html>
<head>
<title>AJAX Demo</title>
<script type="text/javascript" src="jquery.js">
</script>
<script type="text/javascript">
var cnt = 0;
$(function(){
  $.ajaxSettings({
    error:function(){alert("Communication error!");}
  });
  $(":button").click(function(){
    var input = {input:$(":textbox").val(),count:cnt};
    $.getJSON("ajax.php",input,function(json){
      cnt = json.cnt;
      $(".cnt").text(cnt)
      $(".msg").text(json.out);
    });
  });
});
</script>
</head>
<body>
<p>
  Input:
  <input type="textbox"/>
  <input type="button" value="Send"/>
  Output #
  <span class="cnt"></span>:
  <span class="msg"></span>
</p>
</body>
</html>
```

```
<?php
$output = '';

switch($_REQUEST['input']) {
  case 'hello':
    $output = 'Hello back.';
    break;
  case 'foo':
    $output = 'Foo you, too.';
    break;
  case 'bar':
    $output = 'Where Andy Capp can be found.';
    break;
  case 'foobar':
    $output = 'This is German, right?';
    break;
  default:
    $output = 'Unrecognized string.';
}

$count = $_REQUEST['count']+1;

echo json_encode(
  array(
    'out' => $output,
    'cnt' => $count
  )
);

exit;
?>
```



jQuery Resources

- Project website
  - http://www.jquery.com
- Learning Center
  - http://docs.jquery.com/Tutorials
  - http://www.learningjquery.com/
  - http://15daysofjquery.com/
- Support
  - http://docs.jquery.com/Discussion
  - http://www.nabble.com/JQuery-f15494.html mailing list archive
- Documentation
  - http://docs.jquery.com/Main_Page
  - http://www.visualjquery.com
  - http://jquery.bassistance.de/api-browser/
- jQuery Success Stories
  - http://docs.jquery.com/Sites_Using_jQuery